Liz Brondos

Reading Interaction #3:
Conversation between a person and his computer clock

(Clock on the computer turns from 23:59 to 00:00)

Person: Wow, it's midnight already! I wonder how my computer clock knows when it has to change its hour-minute display.
Clock: Would you like to look at my code?
Person: Wow, I didn't know my clock could talk!
Clock: Well, you better believe it! I can try to explain a little bit about how I work and explain some of the code to you, but a lot of it is still a big mystery to me.
Person: That's OK. Just try your best. I took computer science last year, so I should be able to understand most of the things you say. First explain to me the general idea that was used to make your program.
Clock: My programmers used abstraction, which means dividing the problem into sub-problems, and then sub-problems or those sub-problems, etc. They also used modularization, which means dividing a whole problem into well-defined parts that can be looked at separately.
Person: Oh, so it's like "divide and conquer"?
Clock: Exactly! Now let me draw a picture so you can understand better. *(Generates a class diagram and object diagram on the screen.)* This is a class diagram, which shows the classes of an application and the relationships between them. The other one is an object diagram, which shows objects and their relationships at one moment in time while the program is being executed. Within my program there are object types and primitive types.
Person: What are those?
Clock: Variables of object types store references to objects, and primitive types are those that are not object types, such as int, boolean, char, double, long, etc.
Person: Who created all the objects in your program?
Clock: Well, some objects are created by programmers, but some objects are created by other objects by means of the *new* operator. To create a new object using this method, you need to type something like this: new *ClassName (parameter-list).*
Person: How come this class has more than one constructor with the same name?
Clock: A class might have more than one constructor, or it can also have more than one method of the same name, as long as they each have different parameter types. For example, the ClockDisplay class has two constructors. If ClockDisplay(), which has no parameters, is used, then the starting time I display will be 00:00. But if ClockDisplay(hour, minute) is used, then you can set up a different starting time.
Person: How are the methods in your program called?
Clock: Well, in an internal method call, methods call other methods of the same class like this: *methodName (parameter-list).* But in an external method call, methods call methods of other objects using dot notation, like this: *object.methodName (parameter-list).*

Person: You sound like you're speaking Greek, yet some of what you say sounds remotely familiar…
Clock: Good.

(Box pops up on computer, saying "Error! Do you wish to debug?")

Person: Help! My computer has gone berserk!
Computer: No I haven't; I am perfectly sane, thank you very much. This box is called a debugger, stupid.
Person: Wow, my computer can talk too! And it's insulting me! I must be demented!
Computer: Well, I just thought I would enlighten you and tell you what a debugger is. A debugger is a tool that helps to examine the execution of an application. It can find bugs for you.
Person: You mean there are insects in my computer? I need some bug spray!
Computer: There may have been insects inside the Mark II computer, but I don't need to be suffocated with bug spray, all right? Bugs are really just computer errors, not real insects.
Person: Ohhhh, OK. Now I understand!
Computer: Oh, go take another computer science class! I don't want you to be so ignorant that you use me improperly and break me! Honestly, I don't understand what is up with people today…They don't know anything at all about how we computers work; they don't even know what objects and classes are! Seriously, I wish someone would just--- *(Power sleep mode activated.)*